Explainer 2025/02/01

8 Retrieval Augmented Generation (RAG) Architectures You Should Know in 2025

By Conor Kelly
Growth

1. Simple RAG

This is the most basic form of RAG. In this configuration, the language model retrieves relevant documents from a static database in response to a query, and then generates an output based on the retrieved information. This straightforward implementation works well in situations where the database is relatively small and doesn’t require complex handling of large or dynamic datasets.

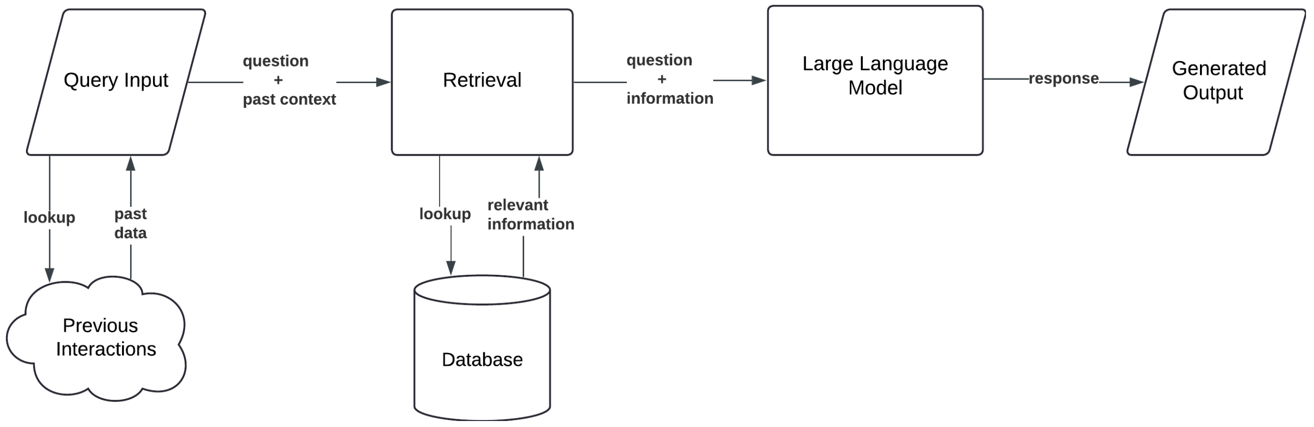
Workflow:

- **Query Input**: The user provides a prompt or question.
- **Document Retrieval**: The model searches a fixed database to find relevant information.
- **Generation**: Based on the retrieved documents, the model generates a response that is grounded in the real-world data it found.

Use Case: Simple RAG is ideal for FAQ systems or customer support bots where responses need to be factually accurate, but the scope of information is limited to a known set of documents, like a product manual or knowledge base.

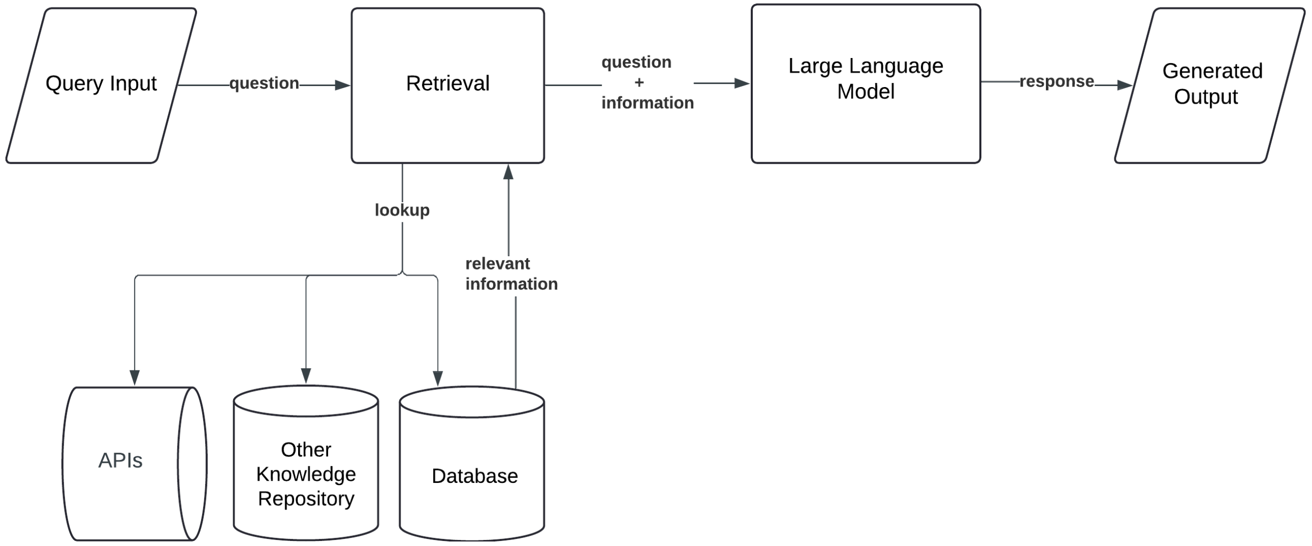
- **Document Retrieval**: It searches the external database for new relevant information.
- **Generation**: The model generates a response by combining retrieved documents with the stored memory.

Use Case: This implementation is particularly useful for chatbots in customer service, where ongoing interactions require the model to remember user preferences or prior issues. It's also beneficial in personalised recommendations where historical data improves the relevance of responses.



Simple RAG with Memory

3. Branched RAG



Branched RAG

4. HyDe (Hypothetical Document Embedding)

HyDe (Hypothetical Document Embedding) is a unique RAG variant that generates hypothetical documents based on the query before retrieving relevant information. Instead of directly retrieving documents from a database, HyDe first creates an embedded representation of what an ideal document might look like, given the query. It then uses this hypothetical document to guide retrieval, improving the relevance and quality of the results.

Workflow:

- **Query Input**: The user provides a prompt or question.

5. Adaptive RAG

Adaptive RAG is a dynamic implementation that adjusts its retrieval strategy based on the complexity or nature of the query. Unlike static models, which follow a single retrieval path regardless of the query, Adaptive RAG can alter its approach in real-time. For simple queries, it might retrieve documents from a single source, while for more complex queries, it may access multiple data sources or employ more sophisticated retrieval techniques.

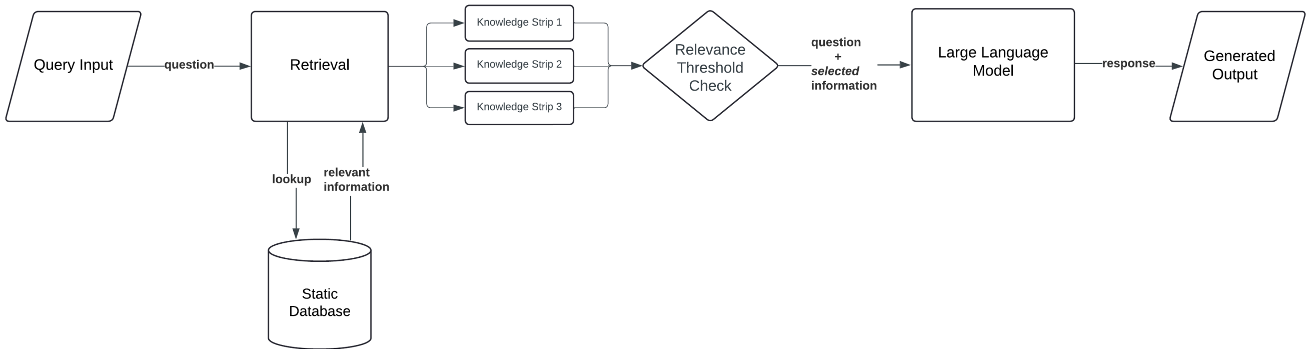
Workflow:

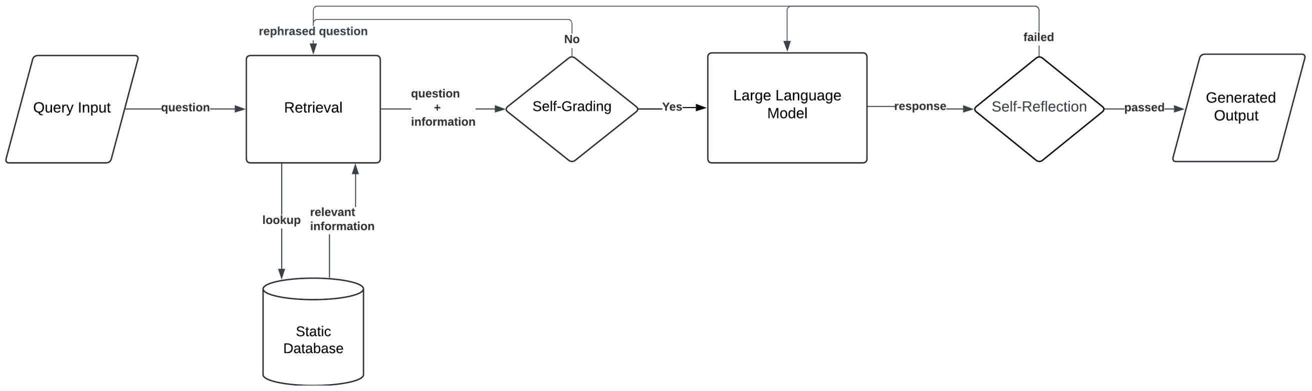
- **Query Input**: The user submits a prompt.
- **Adaptive Retrieval**: Based on the query's complexity, the model decides whether to retrieve documents from one or multiple sources, or to adjust the retrieval method.
- **Generation**: The model processes the retrieved information and generates a tailored response, optimising the retrieval process for each specific query.

Use Case: Adaptive RAG is useful for enterprise search systems, where the nature of the queries can vary significantly. It ensures both simple and complex queries are handled efficiently, providing the best balance of speed and depth.

- **Document Retrieval**: The model retrieves documents from the knowledge base and evaluates their relevance.
- **Knowledge Stripping and Grading**: The retrieved documents are broken down into "knowledge strips" — smaller sections of information. Each strip is graded based on relevance.
- **Knowledge Refinement**: Irrelevant strips are filtered out. If no strip meets the relevance threshold, the model seeks additional information, often using web searches to supplement retrieval.
- **Generation**: Once a satisfactory set of knowledge strips is obtained, the model generates a final response based on the most relevant and accurate information.

Use Case: Corrective RAG is ideal for applications requiring high factual accuracy, such as legal document generation, medical diagnosis support, or financial analysis, where even minor inaccuracies can lead to significant consequences. When evaluating models like CRAG, it's essential to assess both the accuracy of retrieval and the consistency of error correction to ensure optimal performance in sensitive domains.





Self-RAG

8. Agentic RAG

Agentic RAG introduces a more autonomous, agent-like behaviour in the retrieval and generation process. In this implementation, the model acts as an "agent" that can perform complex, multi-step tasks, proactively interacting with multiple data sources or APIs to gather information. What sets Agentic RAG apart is its ability to assign Document Agents to each individual document and orchestrate their interactions through a meta-agent. This system allows for more sophisticated decision-making, enabling the model to determine which retrieval strategies or external systems to engage with based on the complexity of the query.

Workflow:

- **Query Input**: The user submits a complex query or task.

Learn more about building RAG applications

To learn more about RAG, check out our cookbook on **how to evaluate RAG** and our guide on **RAG explained**.

At Humanloop, we make it easy for enterprises to collaboratively develop and evaluate RAG-based AI applications. We provide all the tooling needed to manage prompts and retrievers and evaluate their performance to ensure your RAG application is reliable at scale.

If you're interested in learning how **Humanloop** can enable your enterprise to develop and evaluate RAG applications, **book a demo** to chat with our team.

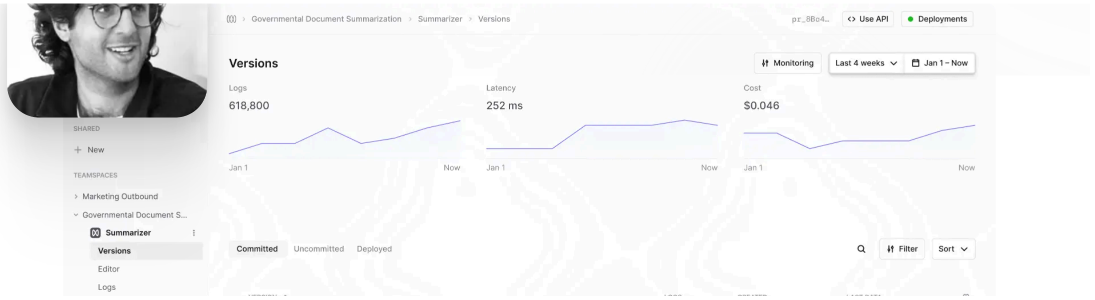
Discuss on X

About the author

Conor Kelly

Growth

Book a demo



Humanloop